REMARKS/ARGUMENTS

Office Action Summary

Status.

1. This *RESPONSE C* is in answer to the Office communication mailed 01/12/2007.
2. The Office communication is not final.
3. NA

Disposition of Claims.

4. Claims 1-3, 5-17 and 19-28 are pending in the application. (Office Action mailed 01/12/2007 Believed In Error as to claims that are pending, See Claim Chart)
5. No Claims have been allowed.
6. Claims 1-3, 5-17 and 19-28 apparently stand rejected. (Office Action mailed 01/12/2007 Believed In Error as to claims that are pending, See Claim Chart)
7. Claims 10, 24 and 26 are objected to.
8. NA
9. NA

Application Papers.

10. NA
11. NA
12. NA

DETAILED ACTION

1. The Examiner indicated that Claim(s) 1-3 and 5-27 have been presented for examination based on the amendment filed on 19th October 2006, but the actual Claims presented for examination are 1-3, 5-17 and 19-28.
2. Applicant's submission filed on 19th October 2006 has been entered.
3. Claim(s) 1, 11, 15 and 25 are Currently Amended with this RESPONSE C and no claims are cancelled.
4. No Claim(s) are added with this RESPONSE C.
5. Claim(s) 1-3 and 5-27 are rejected under 35 USC § 112 and applicant presumes that the Examiner intended that Claims 1-3, 5-17 and 19-28 be rejected under 35 USC § 112.
6. Claim(s) 1-3, 5-9, 11-23, 25 and 27 remain rejected under 35 USC § 103 and applicant presumes that the Examiner intended that Claims are 1-3, 5-9, 11-17, 19-23, 25 and 27-28 are rejected under 35 USC § 103.
7. Claims 10, 24 and 26 are objected to being dependent from rejected claims and may be allowable if presented in independent format after curing the deficiencies of the independent claims.
8. Applicant considers the Examiner's rejections and remarks to apply to Claims 1-3, 5-17 and 19-28.

Examiner's Response (in Office action Dated: 01/12/2007) to Applicant's Previously Filed Arguments In *RESPONSE B* (Filed 10/19/2006)

9. Examiner's Response to Applicant's Section 11.2.1.

9.1. The Examiner's withdrawal of the argument regarding an "indexing table" is noted.

10. Examiner's Response to Applicant's Section 11.2.2

10.1. The Examiner states,

> Applicant seems to be arguing that applicant's indexing table represents species and the TLB (translation look-aside buffer) taught by the prior art represents genus. Examiner had presented the argument that TLB is the species instead, which was acknowledged by applicant in response first office action as being more complex than an indexing table (genus). Examiner maintains the rejection as stated in MPEP 21 31.02 where the species anticipates a genus.
>
> (Response A: Section 10.2 by applicant states in part) ... For example the Examiner must agree. that the indexing table of the applicant could not be used to replace the TLB of the smith 11982 publication. In a similar way, why would anyone adopt the complex structure of TLB, with the large amounts of data implied, to a simple task of a small table in applicant's invention. ..."

10.2. The Examiner misconstrues Applicant's argument. Applicant's argument is simply that **no** generic/species relationship exists! A TLB is **not generic** to Applicant's claimed elements. Applicant's claimed elements are **not generic** to a TLB.

10.3. Claim 1, by way of example and illustration, includes the following **A** and **B** elements:

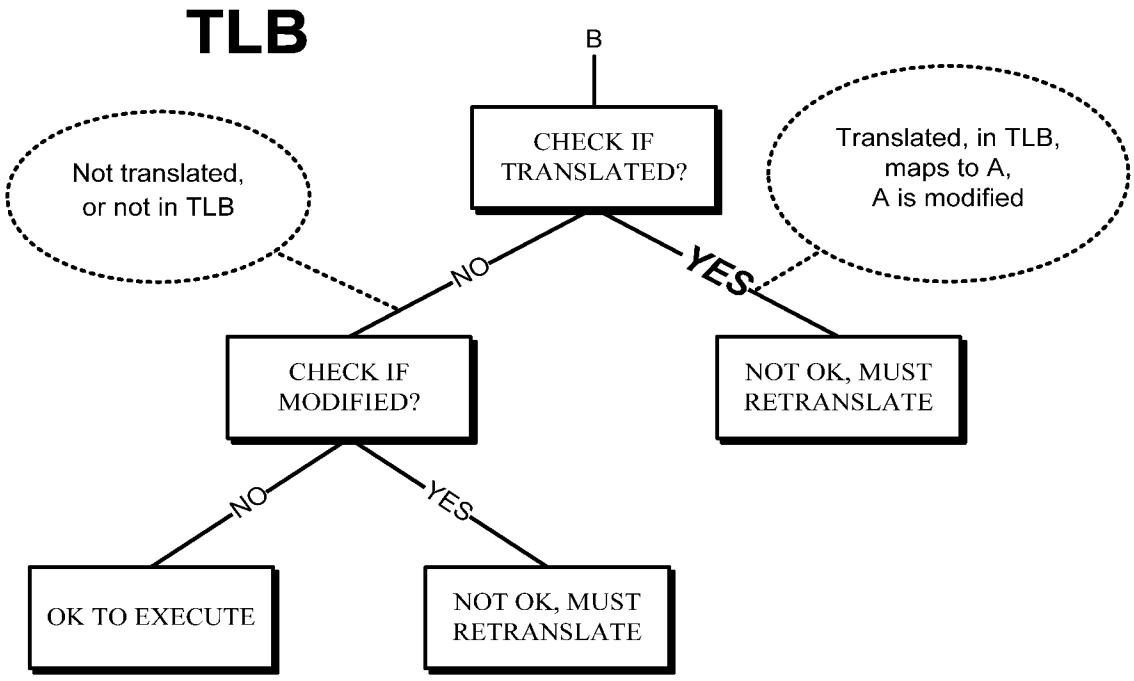
> (**A**) *if the indications indicate that said particular block has not been translated to cause code modification, going directly to said step of executing translated instructions,*
>
> (B) *if the indications indicate that said particular block has been translated so as possibly to cause code modification, checking said translation store to determine if code has been modified and;*

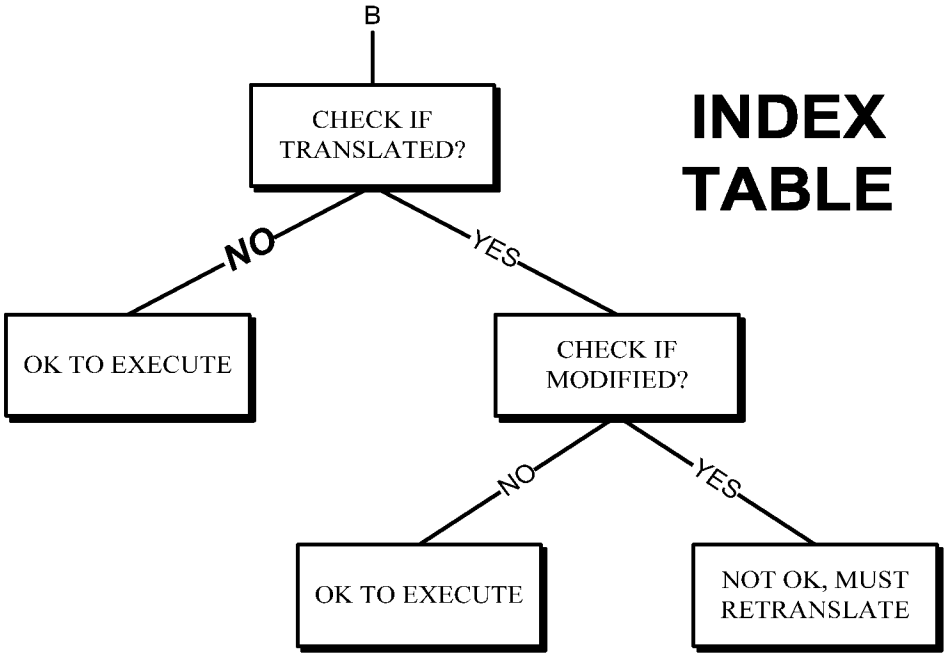
10.4. The **A** and **B** elements as recited in Claim 1 (and in the other independent claims) are claim elements (1) that differentiate the claims of the present application from a TLB, (2) that provide the performance and efficacy of operation and (3) that are novel over any prior art.

10.5. The Examiner admits to confusion, to not finding performance and efficiency recited in the claims and to not understanding why a TLB maximizes the YES path (the path directly to OK TO EXECUTE) and applicant's claimed structure maximizes the NO path (the path NOT directly to OK TO EXECUTE).

10.6. To assist the Examiner in a better understanding of the claimed invention, the claimed "**A** and **B** elements" as identified above **inherently** give rise to the performance and efficiency advantage of the present invention over a TLB structure. The applicant does not believe that those skilled in the art would even consider use of a TLB structure as proposed by the Examiner (because such structure is cumbersome and costly for the purpose suggested by the Examiner). However, because the Examiner has made the suggestion of a TLB, applicant is compelled to distinguish the TLB from Applicant's claimed structure, and the distinction is in the "**A** and **B** elements" as presented in the claims.

10.7. The reason that the "**A** and **B** elements" of the claimed invention inherently give rise to the performance and efficiency asserted by applicant is obvious from inspecting the drawing appearing after paragraph 11.2.2.6 of the *RESPONSE B* dated 10/19/2006 and again reproduced below to avoid the need for switching back and forth between documents.


TLB
B
CHECK IF TRANSLATED?
Not translated, or not in TLB
Translated, in TLB, maps to A, A is modified
NO
YES
CHECK IF MODIFIED?
NOT OK, MUST RETRANSLATE
NO
YES
OK TO EXECUTE
NOT OK, MUST RETRANSLATE


B
CHECK IF TRANSLATED?
INDEX TABLE
NO
YES
OK TO EXECUTE
CHECK IF MODIFIED?
NO
YES
OK TO EXECUTE
NOT OK, MUST RETRANSLATE

10.8. Referring to the TLB of the drawing, efficiency is determined by how fast an OK TO EXECUTE decision is made followed by doing the actual execution. In the TLB operation as seen in the drawing, only one step is used in the YES path to determine that it is NOT OK to execute and hence that MUST RETRANSLATE must be performed. This YES path only uses a single step and hence is efficient in determining NOT OK. However, the efficiency in reaching NOT OK does not constitute efficiency in arriving at OK TO EXECUTE since much additional processing is required after NOT OK before OK TO EXECUTE can be reached. In the TLB, a two step operation, a first NO on the path to CHECK IF MODIFIED followed by a second NO on the path is required in order to arrive at OK TO EXECUTE. This multi-step NO processing of the TLB is inherently much less efficient than the one step YES processing of the TLB and that is why the TLB structure inherently is said to maximize the YES processing (that is, the TLB maximizes the path **not leading** directly to OK TO EXECUTE).

10.9. Referring to the INDEX TABLE of the drawing, efficiency is again determined by how fast an OK TO EXECUTE decision is made followed of course by doing the actual execution. In the INDEX TABLE of the drawing, a single NO step is required to arrive at OK TO EXECUTE and hence applicant's INDEX TABLE is very efficient in arriving at an OK TO EXECUTE. The structure of the INDEX TABLE is inherently much faster in processing the single NO path than the multi-path YES processing and that is why the INDEX TABLE is said to maximize the NO processing (that is, maximizes the path **leading** directly to OK TO EXECUTE).

10.10. The Examiner argues that there is "no limitation presented in the claim that indicates maximizing the YES path and NO path or the intended difference in performance & efficiency of the indexing table over a TLB". The Examiner is believed in error in arguing that there are no limitations in the claims that pertain to the difference in performance and

efficiency of the INDEX TABLE over the TLB. The recitation of the "**A** and **B** elements" are precisely the limitations in the claims that result in the difference in performance and efficiency.

10.11. In summary, Applicant's argument is that no suggestion anywhere and no one of ordinary skill in the art would use a TLB as proposed by the Examiner because a TLB is too costly and cumbersome. However, for the sake of argument, even if a TLB were employed as suggested by the Examiner, the TLB still would not have the claimed "**A** and **B** elements" of Applicant's claims and accordingly would not have the performance and efficiency inherent in Applicant's index structure.

11. Examiner's Response to Applicant's Section 11.2.3

11.1. The Examiner has shifted position and now argues in the 01/12/2007 Office action based upon ***block table entries***:

> *Applicant is arguing that Mann '295 is does not disclose reuse of the code by other* ***block table entries*** *(individual instructions) as asserted by examiner.*

11.2. Previously, the Examiner had argued based upon ***other block entry tables***:

> *The reuse of the code by* ***other block entry tables*** *requires a many to one or one to many mapping between the translated code in the host code block and block entry tables. This deficiency is not addressed in Mann '295 and would be necessary for efficient code reuse (Problem to be solved).*

11.3. The difference between ***block table entries*** in the 01/12/2007 Office action and ***other block entry tables*** in the prior 05/18/2006 Office action is significant. Applicant's invention permits different blocks of legacy code to be mapped to the same block of the trans-

lated code (this block mapping is a many-to-one block mapping). *Mann '295* does not have such many-to-one block mapping.

11.4. The Examiner now switches to discussing the functioning of the ***block table entries*** in a one-to-one block mapping environment. Applicant agrees with the Examiner's new argument to the extent that different legacy instructions 76 in a single block of such legacy instructions in *Mann '295* are mapped to different addresses in the single block of Host Code 88 by the Tables 80. Such one-to-one mapping of Tables 80 is described as a simple linking list and there is no reason to believe that anything more complex, such as a TLB, is needed, is practical, is possible or would have any advantage. There is no identification anywhere of any problem or other need that would suggest to anyone of ordinary skill in the art that a TLB should be added to *Mann '295.*

11.5. In *Mann '295*, a block is defined as a number of target code instructions with the first target instruction of the block marked by an "F" and the last target instruction of the block marked by an "L" as shown in FIG 3 of *Mann '295* for a single target block. The single target block of FIG 3 (all the target instructions between "F" and "L") has each target instruction translated to one or more host instructions in the single block of Host Code 88, that is, *Mann '295* is a one-to-one target block to host block mapping. Further, *Mann '295* is a one-to-one mapping for each target instruction in a single target block to host instruction(s) in the single host block. Nothing in *Mann '295* suggests that any problem exists whereby host instructions for any one target instruction (in one target block) would map to the same address space of host instructions(s) for some other target instruction (in the one same target block). In *Mann '295,* the mapping done by Tables 80 is simple and presents no problem to be solved.

11.6. Nothing in *Mann '295* describes an operation wherein for some first target instruction and some second target instruction (either from a single target block or from multiple target blocks), the first and second target instructions are mapped to first host instructions and second host instructions in a host code block wherein the host code address spaces for the first host instructions potentially overlap or otherwise conflict with the host code address spaces for the second host instructions. Rather, *Mann '295* merely describes (1) a one-to-one single target block to single host block structure and (2) a one-to-one single target instruction to a single set of one or more host code instructions for emulating the single target instruction. Nothing in *Mann '295* suggests any operation wherein a host code address conflict might arise from a many-to-one mapping.

11.7. The *Smith* reference relied upon by the Examiner contemplates a many-to-one mapping and uses a TLB to resolve address conflicts in such a structure. Since *Mann '295* does not describe any many-to-one structure or operation, there is no reason to combine the *Smith* TLB with *Mann '295.* The only reason to make the combination is the Examiner's hind-sight reliance on Applicant's invention.

11.8. By way of further distinction, Applicant's operation permits of a many-to-one "target block"-to- "host block" mapping whereby "host block" addresses from different "target block" instructions can conflict. Applicant's claims, including the "**A** and **B** elements" as previously described provide a novel manner of processing in the many-to-one environment. *Mann '295* does not have the many-to-one environment and hence combining the *Smith* reference with *Mann '295* does not solve any problem and would certainly not be considered by one skilled in the art.

12. Examiner's Response to Applicant's Section 11.2.4

12.1. Pursuant to the Examiner's suggestion, the independent claims have been amended to positively recite that instructions with "self-modifying code" are included. The claims as previously presented, of course, contemplated self-modifying code since the claims operate to check whether such self modifications have been made.

12.2. The Examiner's comments about the Wikipedia definition of self-modifying code have been noted.

12.3. The Examiner requested that applicant identify in the specification where self-modifying code is described. Applicant's specification, paragraphs [39] and [40] as filed, refers to the difference between "code" modification and "data" modification as follows:

> The TABLE is checked (arrow 4 in FIG. 4) after the store has been performed (arrow 3 in FIG. 4) by execution of the CMP instruction to make sure that the translation of the victim block (the second block in this case) cannot occur after the check (CMP) but before the store (ST4). Likewise, the incrementation of the TABLE entry is done at the beginning of the block translation process (arrow 2 in FIG. 4), before the victim instruction (OLDI) has been fetched. Both of these conditions are met to insure that a potential code modification will not be missed using the tracking table 22. If a potential code modification is detected (that is, the TABLE entry is not 0), then the CHK_MODIFIED routine is called by the BNE instruction to check in the translation store 24 to determine whether any code was actually modified.
>
> In the FIG. 4 example, for simplicity of explanation, the CISC blocks do not contain both instructions and data. In the case where a CISC block does contain both instructions and data, storing to the data portion does not constitute actual instruction code modification. In the FIG. 4 example, actual code was modified, so the victim block is invalidated (arrow 5 in FIG. 4). It is then re-translated as before, with the SLA (NEWI) having replaced the SH (OLDI).

12.4. To clarify the operation of the claims, the independent claims have been amended to recite "code modification" to designate the operation that occurs when self-modifying code causes a code modification.

12.5. The terminology "code modification" is believed to be clearer and has been presented in the claims by amendment pursuant to the Examiner's comments and consistent with the terminology used in Applicant's specification as identified in the quote above.

Examiner's Remarks for 35 U.S.C. § 103

13. Examiner's Response to Applicant's Section 12.1

13.1. No comment required.

14. Examiner's Response to Applicant's Section 12.2

14.1. As explained above, Applicant's invention permits different blocks of legacy code to be mapped to the same address space of a single block of translated code (a many-to-one block operation) whereas *Mann '295* has no such description and is only a one-to-one block operation. In this block sense, Applicant's operation is many-to-one whereas *Mann '295* is only one-to-one.

14.2. The Examiner further argues,

> Further, Mann '295 clearly teaches reuse of the translated code which implies many (legacy instructions) to one (host code); thereby teaching block numbers in the said table are the same for multiple different blocks. Further, same index-offset information is stored (block address digits) for instructions in the same block (See Mann '295 : Fig.3 element 74). Also see Mann '295 : Fig.4; Co1.9 Lines 10-36, Co1.7 Lines 49- Co1.8 Line 65.

14.3. Applicant agrees with the Examiner that many target instructions originate in one target block. Additionally, those many target instructions in one block are mapped to many host instructions in the host code block. Applicant does not agree that such a one-to-one relationship implies that "block numbers in the said table are the same for multiple differ-

ent blocks". Each target instruction is mapped by Tables 80 to a unique block address for corresponding host instructions that emulate the target instruction so that no block numbers are the same for multiple different blocks.

14.4. Applicant believes that the Claims, particularly as amended, exclude "interpretation" of target instructions. None of the steps in any of applicant's claims relate to interpretation of instructions. All operations in all of Applicant's claims are with respect to translated instructions.

15. Examiner's Response to Applicant's Section 12.3-12.6.

15.1. Applicant's prior arguments are clear and are reasserted here.

16. Examiner's Response to Applicant's Section 12.7

16.1. The Examiner asserts that FIG 4 of *Mann '295* depicts something about subsets, but Applicant cannot find anything in FIG 4, or in the text describing FIG 4, that supports the Examiner's argument. The Examiner is requested to identify what in FIG 4 or otherwise the Examiner is relying on to show subsets.

16.2. The Examiner's comment regarding allowable subject matter is noted.

17. Examiner's Response to Applicant's Section 12.8

17.1. No comment required

18. Examiner's Response to Applicant's Section 12.9

18.1. Applicant's previous arguments are maintained.

19. Examiner's Response to Applicant's Section 12.10

19.1. None of the operations in Applicant's claims are related to interpretation and all executions in Applicant's claims are done by execution of translated code as is clear from Applicant's specification. Accordingly, the Examiner's argument that Applicant's claims cover "interpretation" is not understood and not believed accurate. Applicant believes the amended claim language further clarifies the operation to the Examiner.

20. Examiner's Response to Applicant's Section 12.11

20.1. Applicant's arguments are previously presented in response to the 12.7.4.1 and 11.2.3.1 sections are maintained.

Examiner's Response to Applicant's Section 12.11, 13, 14 and 16

20.2. Applicant maintains the argument's previously presented in light of the amendments to the claims.

Claim Rejections - 35 USC § 112

The quotation of the second paragraph of 35 U.S.C. § 112 is noted.

21. Claims 1-3 and 5-27 stand rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

21.1. Claims 1, 11, 15 and 25 have been amended to recite, for example, "…has not been translated to cause code modification…" and therefore Claims 1, 11, 15 and 25 and all claims dependent there from are not now believed to be indefinite.

Claim Rejections - 35 USC 8 103

The quotation of 35 U.S.C. 103(a) is noted.

This application DOES NOT name joint inventors.

22. Claims 1-3, 5, 7-9, 11-13,15-17, 19, 21-23,25, and 27 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6516295 issued to George A. Mann et al (Mann '295 hereafter), further in view of ACM Article "Cache Memories" by Alan Jay Smith (Smith '1982 hereafter).

22.1. Regarding Claim 1, the Examiner applies the rejection in the present Office action dated 01/12/2007 in a manner identical to the way the Examiner applied the rejection in the previous Office action.

22.2. Applicant reasserts and incorporates herein the traversal of those arguments as made in *RESPONSE B* (filed 10/19/2006) to the previous Office action.

22.3. The Examiner's arguments directed to Claim 1 (as might be applied to Claim 1 as amended) is not supported by *Mann '295* because the Examiner fails to distinguish between operations that are "interpretation" versus operations that are execution of "translated instructions". This distinction is made more clear in the amendments to the claims presented in this *RESPONSE C*. In order to understand the differences in light of the newly amended claims, a review of the operation of *Mann '295* is first warranted.

22.3.1. The general operation of *Mann '295* is depicted in FIG. 4. The commencement of operation is described in *Mann '295,* Col: 5 Line 64, where a Target instruction 76 is first interpreted by setting the code tag 72 to "C". Hence in the first iteration of FIG 4 after START will have the Tag="F", Tag="E", Tag="M", Tag="L" conditions are all "No" thus arriving at the Tag="C" test 106. Since in the first iteration the Tag has just been set to "C", the 106 test result is "Yes" and the counter is

incremented in 114. Since the first operation is being described, the "Count>Max?" test of 122 result is "No" and hence the instruction is interpreted in 128. If it is determined that the instruction is self-modifying code, then special handling is required (see Col: 7, Lines 4-37). As a result of such special handling, the Target code 72 is set to "X" as described in FIG 5. Thereafter, the processing in FIG. 4 with Target code 72 set to "X", has all of the Tag="F", Tag="E", Tag="M", Tag="L", Tag="C" conditions "No" so that all processing reverts to the "Interpret Instruction" 128 mode of operation.

22.3.2. The specific operation of *Mann '295* treats self-modifying code in two ways. In the first way, the entire block containing the self-modifying instruction is marked with an "X" and hence all processing of that block thereafter is by "interpretation" through test 108 and interpretation in 108 in FIG 4 and not by execution of "translated instructions". In the second way, a first portion of the block (forming a "new first block") up to and not including the self-modifying instruction has the last entry marked with an "L" and processing is transferred to "interpretation" for self-modifying instructions. For a second portion of the block (forming a "new second block"), after and not including the self-modifying instruction, is marked with an "F" and is processed by executing "translated instructions" unless, of course, another self-modifying instruction is encountered in the "new second block" in which case the transfer to "interpretation" is again repeated. The only manner of processing self-modifying instructions in *Mann '295* is by "interpretation" and not by execution of "translated instructions".

22.3.3. By way of distinction in the present invention, each translated self-modifying instruction is processed by execution of "translated instructions" and not by interpretation. The execution of "translated instructions" in the present invention occurs con-

ditionally as a function of whether or not prior execution of such "translated instruction(s)" resulting from a self-modifying instruction has or has not actually resulted in a modification of code. If no modification of code has occurred in the present invention, the "translated instruction(s)" resulting from a self-modifying instruction are executed directly and efficiently without further processing (Distinguish *Mann '295* which always wastes time by reverting to "interpretation" for self-modifying instructions). If modification of code has occurred in the present invention, the "translated instruction(s)" resulting from a self-modifying instruction are retranslated and then the retranslated instructions are executed directly. The differences between *Mann '295* and the present invention are several. *Mann '295* always reverts to "interpretation" when a self-modifying instruction is encountered and the present invention never need revert to "interpretation". *Mann '295* always has a time penalty in that upon encountering a self-modifying instruction, *Mann '295* always reverts to more time-expending "interpretation". In the present invention, a time penalty is experienced only under the condition that "translated instruction(s)" resulting from a self-modifying instruction has(have) actually resulted in a modification of code. The statistical frequency of occurrence of self-modifying instructions that **do** actually result in a modification of code in many applications is much lower than the statistical frequency of occurrence of self-modifying instructions that **do not** actually result in a modification of code. Accordingly, on a statistical basis, the present invention which does not require retranslation in the absence of a modification of the code is far superior to any operation in *Mann '295* that always reverts to time-wasting "interpretation".

22.3.4. Referring specifically to the Examiner's rejection of Claim 1, the Examiner ignores the part of the claim that recites "*said storing translation indications using a subset of block address digits whereby block numbers in said table are the same for*

multiple different blocks,". The Examiner ignores this recitation of *multiple different blocks* in Applicant's Claim 1 because *Mann '295* is a one-to-one structure that neither requires nor permits *multiple different blocks*. The Examiner does not identify anything in *Mann '295* that corresponds to Applicant's recitation "*said storing translation indications using a subset of block address digits whereby block numbers in said table are the same for multiple different blocks,*". At least for this reason, the Examiner's rejection cannot be sustained.

22.3.5. Referring specifically to the Examiner's statement "*going to the step of executing the translated instruction*" is erroneous in that no execution of a "translated instruction" occurs, but rather as is clear from the Examiner's citation *(Mann '295: Fig. 4 Path 108, 128, 129),* processing reverts in *Mann '295* to a time-wasting "interpretation" in "Interpret Instruction 128" as shown and described in connection with FIG 4 of *Mann '295*. At least for this reason, the Examiner's rejection cannot be sustained.

22.3.6. Referring specifically to the Examiner's rejection to the extent that the Examiner suggests that in *Mann '295* "translated instruction(s)" from a self-modifying instruction are retranslated and then executed is in error. Unlike the present invention, a self-modifying instruction in *Mann '295* always reverts to a time-wasting "interpretation" rather than more efficient processing to execute a "translated instruction". The Examiner ignores the fact that the processing of the "translated instructions" of the original block in *Mann '295* is carried out by partitioning the original block into two smaller blocks. The partitioned two smaller blocks in *Mann '295* include a "new first block" including instructions **up to and not including the original self-modifying instruction** and a "new second block" including instructions **after** the "translated instruction(s)" for the self-modifying instruction. The processing in *Mann*

'295 of the self-modifying instruction is only done, contrary to the Examiner's argument, by a time-wasting "interpretation" and is not done by a retranslation and then execution of the retranslated instructions. At least for this reason, the Examiner's rejection cannot be sustained.

22.4. In making the rejection of Claim 1, the Examiner further argues,

Mann '295 does not teach the details of the limitation presented above explicitly. Smith '1982 teaches that TLB having a hashing mechanism to map the virtual addresses (block numbers) to the real address (translated host code block) (Smith '1982: Pg.475 Co1.2 Paragraphs 3-4). Hashing (done by taking an XOR or through randomized algorithm) depends on the number of bits selected, resulting in folding or overlapping (Smith '1982: Pg.488, Col.1 Paragraph 1 ; Pg.489, Col.1). The hashing scheme selected by the applicant is extremely simplified version, as only one middle bit is selected to index the translation indication index table.

It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to combine the teachings of Smith '1982 and apply them to Mann '295 to implement the indexing table as disclosed. The motivation would have been that Smith '1982 discloses the necessity for TLB like lookup when dealing with translated information when address space doesn't map directly (Smith '1982: Pg.510, Section 2.9, 2.17). Mann '295's design requires translation as one target code block may have one or more translated host code instructions. Hence Smith '1982 solves Mann '295's problem of mapping the target legacy instruction object to translated host object code block (Mann

'295: Co1.6 Lines 47-55). Please see further the response to arguments for clarification.

22.4.1. The arguments of the Examiner as quoted in the previous section immediately above are formulated by the Examiner only after reading and finding a suggestion therefor from Applicant's specification. The Examiner has admitted that one skilled in the art would not equate a simple indexing table with a TLB. Furthermore, the indexing table of *Mann '295* requires a one-to-one indexing and changing to a many-to-one operation, as suggested by the Examiner, would destroy the simplicity, lower cost and intended one-to-one operation of *Mann '295.* Such modifications to *Mann '295* would render *Mann '295* inoperable because there are no facilities in *Mann '295* for processing in a many-to-one environment. The suggestion for modifying *Mann '295* to be and operate like Applicant's invention can only be found in Applicant's specification. Certainly, one skilled in that art would find no reason and certainly no suggestion for changing the simple and inexpensive one-to-one operation of *Mann '295* to a very complex TLB many-to-one operation that provides no benefit to *Mann '295.* The Examiner only finds such motivation from Applicant's specification; there are no legitimate technical reasons why any one skilled in the art would choose to modify *Mann '295* in the manner suggested by the Examiner.

22.5. Furthermore, even if *Mann '295* were modified to include a TLB as suggested by the Examiner, the operation would not be the same as the index table and operation claimed by Applicant. Specifically, even if modified as suggested by the Examiner, the combination of *Mann '295* and *Smith* would not include the previously described "**A** and **B** elements" as folows:

(**A**) *if the indications indicate that said particular block has not been translated to cause code modification, going directly to said step of executing translated instructions,*

(B) *if the indications indicate that said particular block has been translated so as possibly to cause code modification, checking said translation store to determine if code has been modified and;*

23. The arguments above are directed to Claim 1 as representative of all claims and the arguments are also applicable and presented for all of the rejected claims for the same reasons. Accordingly, reconsideration of all rejected claims is respectfully requested.

24. The Examiner has again rejected each of the remaining claims for the same reasons presented in the previous Office action and Applicant's arguments traversing those rejections in the previous *RESPONSE B* are incorporated by reference herein in this *RESPONSE C*. Additionally, Applicant's arguments should be construed in light of the amendments to the claims made in this *RESPONSE C*.

Allowable Subject Matter

25. Claims 10, 24 and 26 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

26. The Examiner has indicated the existence of allowable subject matter.

26.1. The Examiner's statement of reasons for the indication of allowable subject matter are as follows:

> Mann '295 does not teaches (sic) incrementing/decrementing the count for the modified/removed block, but teaches equivalent functionality of removing and de-allocating the block entry table (indexing table entry) when the translated host block is no longer needed/garbage collected (Mann '295: Co1.7 Lines 62-67, Co1.8 Lines 1-3). Mann '295 also teaches that the translation indication includes a state field (tag field), for each block number, indicating the block has been modified (Mann '295: Co1.5 Table I; Co1.6 Lines 62-67). Further, Mann '295 teaches incrementing the state field each time block is executed on the cache (Mann '295: Co1.6 Lines 64-65). Therefore the count is not associated to the addition or removal of block.

26.2. Applicant agrees to the existence of allowable subject matter, but disputes portions of the Examiner's statement as quoted above that represents a statement of reasons for the indication of allowable subject matter. Specifically, *Mann '295* operates to determine the presence of self-modifying code and does not determine whether any code modification actually occurred. Rather, upon detection of self-modifying code, *Mann '295* aborts to "interpretation" and to the extent that the Examiner suggests otherwise in the statement of reasons for the indication of allowable subject matter, the Examiner is in error.

Conclusion

27. All claims stand rejected and Applicant believes that upon the reconsideration requested by Applicant and in light of the amendments to the claims, the Examiner will now find all claims allowable.

28. The prior art made of record and not relied upon is noted.

29. The **Examiner's Note** to the effect that that the Examiner has cited particular columns and line numbers in the references is acknowledged. Applicant has examined the references and cannot, in many instances, find support for the Examiner's arguments. Accordingly, if the Examiner is aware of any citations that support the Examiner's arguments, the Examiner is requested to make the citations of record.

30. In view of the above remarks, reconsideration of all claims in the application is requested.

Respectfully submitted,

	SIGNATURE	
David E. Lovejoy (US Reg. No.: 22,748)	David E. Lovejoy **/david lovejoy/**	**Signature Date** 11 March 2007
Mail Address	Customer No.	Communication
102 Reed Ranch Rd. Tiburon, CA 94920-2025 USA	21603	Tel: (415) 435-8203 Fax: (415) 435-8857 e-mail:david.lovejoy@sbcglobal.net